

Aufgabe 1

Ändern Sie die Syntax von WHILE, indem Sie INTEGER- und REALZahlen unterscheiden.
In der Vorlesung wurden die ganzen Zahlen wie folgt definiert:

```
1 // ganze Zahlen (endlicher Ausschnitt der ganzen Zahlen MIN+1 .. MAX)
2 Z ::= 0 | 1 | ... | MAX | -1 | -2 | ... | MIN
```

Gleitkommazahlen können in WHILE so abgebildet werden:

```
1 // natürliche Zahlen mit 0
2 N ::= 0 | 1 | ... | MAX
3 // reelle Zahlen
4 R ::= Z.N
```

Die Definition der reellen Zahlen muss zur Vollständigkeit in die Konstanten und Terme eingetragen werden.

```
1 K ::= Z | R | W
2 T ::= Z | R | I | T1 OP T2 | read, für T1, T2 in TERM
```

Aufgabe 2

Definieren Sie für eine geeignete Erweiterung der Sprache WHILE eine konkrete Syntax, die eindeutig ist.
Zusätzlich zum **if else** wäre **switch** ganz praktisch:

```
1 C ::= switch T: CASE // dazu kommen die anderen Definitionen aus der VL.
2 CASE ::= CASE CASE | case B do C | default C
```

Man könnte beim CASE noch ein einfaches skip einfügen. Dann hat man ebenfalls das Standardverhalten von C, wenn kein **default** angegeben wurde.

Eine weitere (, triviale, aber) wirklich(!) sinnvolle Erweiterung wäre das *Programm*. Es besteht aus einem oder mehreren Befehlen. Weil die Menge der Befehle bereits ausreichend definiert ist sieht diese Erweiterung ziemlich unspektakulär aus:

```
1 //Programm
2 P ::= C
```

Aufgabe 3

Formulieren Sie informell eine Präzisierung der angegebenen WHILE-Semantik, die die genannten Fehlerquellen behandelt.

Bereichsüberschreitungen Zwei Fälle:

1. Man verlässt z.B. den definierten Bereich der ganzen Zahlen, also MIN-1 oder MAX+1:
Nachfolger von MAX ist MIN, Vorgänger von MIN ist MAX. Ergebnis ist, dass man damit das aus z.B. C gewohnte Integer-Overflow-Verhalten bekommt.
2. Über das Ende einer Zeile oder einer Datei hinaus lesen:
read sollte \r\n für Zeilen oder EOF für Dateien erkennen und je nach Spezifikation nach Parsen dieser Flags beendet werden. Existieren diese Steuerzeichen nicht, so muss read abbrechen. Abbruchbedingungen müssen in diesem Fall genauer spezifiziert werden. Das Verhalten beim Abbruch könnte ähnlich dem einer leeren Eingabedatei sein oder read gibt im Fehlerfall eine -1 zurück.

Division durch Null Wenn OP in T1 OP T2 die Division ist, dann darf T2 keine Null sein.

Da Division durch Null nicht definiert ist, wird ein Fehler geworfen und die Programmausführung abgebrochen.

Berechnung von read bei leerer Eingabedatei Im Falle einer leeren Eingabedatei liest `read` für `B` `false` und für `Z` `0`.

Typkonflikte Es muss zunächst geprüft werden, ob der Typ von `T1` mit dem Typ von `T2` kompatibel ist. Darüber hinaus muss die Operation `OP` auf `T1` und `T2` definiert sein. Weitere Möglichkeiten: wird ein `B` erwartet, aber ein `T` oder ein `Z` gelesen, wird `0` als `FALSE` und alles andere als `TRUE` verstanden (analog zur C Semantik). Alternativ führt jeder Typfehler zu einem Fehler und das Programm wird an der Stelle des Fehlers abgebrochen.